Time-Varying Graph Patrolling Against Attackers with Locally Limited and Imperfect Observation Models

Carlos Diaz Alvarenga Nicola Basilico Stefano Carpin

***Abstract*— The use of autonomous robots for surveillance is one of the most interesting applications of graph-patrolling algorithms. In recent years, considerable effort has been devoted to tackling the problem of efficiently computing effective patrolling strategies. One of the mainstream approaches is adversarial patrolling, where a model of a strategic attacker is explicitly taken into account. A common assumption made by these techniques is to consider a worst-case attacker, characterized by ubiquitous and perfect observation capabilities. Motivated by the domain of robotic applications, we instead consider a more realistic and limited attacker model capable of gathering noisy observations in a locally limited range of the environment. We assume that the modeled attacker follows a behavior induced by its observations. Thus, we devise a randomized patrolling strategy based on Markov chains that makes observations reveal very little information, while still maintaining a reasonable level of protection in the environment. Our experimental results obtained in simulation confirm time-variance as a practical approach for our objective.**

I. Introduction

With increasing levels of intelligence and automation mobile robots are now an enabling technology for autonomous patrolling of indoor and outdoor environments [16]. Patrolling is a repetitive and potentially dangerous task whose execution costs can be mitigated by deploying surveillance robots in the area of interest. Because these systems are designed to be autonomous, the high-level planning of the surveillance activities emerges as one of the most critical challenges to achieving good performance and, ultimately, detecting and preventing malicious activities in the environment. Research in multi-agent and multi-robot systems has produced decision-making algorithms for this problem. Issues like how to plan efficient paths, where and when to schedule surveillance actions, and how to coordinate with teammates have been tackled by models encoding some environment representation (for example, continuous or discrete) and assumptions on agents capabilities and behaviors.

Generally speaking, this sub-field moved from single-agent optimization models to more sophisticated adversarial settings. The effort has constantly been that of capturing more realistic situations to better model real-life applications, and consequently provide more effective solutions. With this perspective, models have been developed to include features such as cost measures, uncertainties, the presence of adversaries with bounded rationality or agents with limited knowledge available to them (see our review of related work in Section II).

In this work, we concentrate on adversarial patrolling on a graph with a single mobile robot and a single attacker. We start from a common graph-based patrolling setting [7], and extend it by introducing an attacker model whose ability of acquiring strategic knowledge is restricted. Specifically, we assume an attacker that can access only a local view of the the environment and its features, and that can only make local observations of the patrolling robot. These assumptions capture situations where acquiring strategic information about patrolling activities is hard. This could be due, for example, to the impossibility of gaining multiple and concurrent observation spots in a cluttered or vast environment, or to non-affordable costs for conducting widespread intelligence activities before deciding where and when to attack. We focus on the problem of ensuring protection to the environment while, at the same time, hindering as much as possible the process with which the attacker propagates a belief from the collected observations with the aim of evaluating the probability of success for a potential attack.

Our contributions build on our previous work, discussed in [6], where we introduced a model for patrolling against a local observer and we provided the first solution based on the idea of strategically injecting delays in the paths followed by the robot (see Section IV for a summary). With respect to such work, here we extend the basic model by introducing observation errors in the attacker model, proposing the use of time-variant Markov strategies to decrease the level of correlations in the sequence of observations made by the attacker (hence making it more difficult to construct a belief to exploit), and conducting extensive experimental tests against a broader set of learning strategies used by the attacker.

II. State of the art

The use of mobile robots for surveillance is one of the flagship applications of the field attracting a significant amount of research [20]. Here, the central planning problem is that of computing a *patrolling strategy* for orchestrating, in space and time, the surveillance activities of a robot.

A first formalization and analysis of this problem in multi-agent terms has been proposed in [10]. This, and other seminal works dealing with patrolling on a graph-represented environment, framed the search for effective patrolling strategies as optimization problems where *idleness*-based metrics were adopted as initial candidate objective

C. Diaz Alvarenga and S. Carpin are with the Department of Computer Science and Engineering, University of California, Merced, CA, USA.
N. Basilico is with the Department of Computer Science, University of Milan, Italy.
C. Diaz Alvarenga is partially supported by the NSF under grant DGE-1633722. Any opinions, findings, conclusions, or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the NSF.

functions. The rationale behind this class of works is the optimization of the inter-visit delays on some vertices of the graph, each under specific modeling and operational assumptions. The main limitation of these contributions is the fact that they neglect the adversarial nature of many patrolling problems which often model the presence of a rational observer who can learn how patrolling is carried out, predict the next moves up to some uncertainty, and devise a best attacking response that takes into account such knowledge. Despite such shortcomings and the availability of more sophisticated methods dealing with it (see below), these techniques still enjoy widespread use in real-world implementations of robotic patrolling systems [22].

Security games [24] introduced game-theoretical models for strategic resource allocation in the presence of adversaries. Robotic patrolling can be seen as one of these problems where the resource to allocate is a robot moving on a graph while the adversary (attacker) tries to compromise some vertex. One common assumption is to adopt a *Leader-Follower* interaction model [11], where the patrolling strategy is common knowledge since the attacker can observe its execution for an arbitrary amount of time. The approach prescribes to commit to the best patrolling strategy given that the attacker will best respond to it.

The Leader-Follower assumption implies full observability of the environment and unlimited observation capabilities of the attacker, two features that properly define a worst case scenario but that are rarely satisfied in reality. A number of works challenged these assumptions, like we do in this work. In [2], the case of perimeter patrolling is considered. In such work, different degrees of knowledge possessed by the attacker are analyzed, from a zero-knowledge attacker to a fully informed one and some resolution methods are proposed and compared. However, the work considers gathered knowledge from a general point of view without adopting an explicit observation model for the attacker. Similarly, other works have investigated the presence of noise in the patrolling strategy observed by the attacker and devised robust resolution methods. One example has been proposed in [19] where bounded rationality is also considered for the attacker. A similar approach is adopted in [26] where the authors assume that the attacker cannot always observe the current allocation of patrolling resources in the environment. In [4] the attacker is allowed to perform a limited number of observations from which it can propagate a belief over the patrolling strategy, while in [25] the patrolling strategy is assumed to be known but the attacker has no access to its real-time realization (that is, it cannot assess the protection status of the targets). This last work introduces leakage as a way to episodically obtain such a knowledge from the target of interest. Other examples are found in [3] where the attacker constructs a belief of the patrolling strategy by performing costly observations and dealing with a trade-off between cost of observation and risk of capture. Finally, in [8] the authors consider a similar belief-based observing attacker and show that planning against the strongest observer might induce limited losses of utility.

The approaches discussed above cast the limited observation capabilities of the attacker to some degree of uncertainty in the knowledge it considers in computing its best response. However, even if affected by errors or sometimes not accessible, observations are assumed to entirely span an environment whose structure is known by the attacker (with the exception of [25] where the patrolling strategy is assumed to be known). In this work, we adopt a *local* observation model that allows the attacker to collect information only for a single target. As a consequence, the environment and the patrolling strategy are always unknown while its realization can only be accessed at a single and fixed vertex. Attacker's locality has been investigated in [5], but not in relation to the observation process. Instead, it is exploited in fixing one attacker behavior according to when it positions itself at a target and then, exploiting a local view, starts its attack as soon as the patroller leaves for other targets.

Markov chains where states encode the current patroller's location have been extensively used to encode patrolling strategies on graph-represented environments (see, for example, [12], [1], [7]) despite their poor scalability when extending states to a history of previous visits [7] (a limitation recently addressed in [15] where response to sequential attacks is studied) and despite their possible failure to describe the optimal patrolling strategy [14]. To the best of our knowledge, the use of time-variance in the transition matrix to deal with an observing attacker has never been investigated.

III. Problem setting

We adopt a patrolling model built on a customary setting that has been adopted in numerous works dealing with patrolling robots (see, for example [7]). We consider a discretized environment consisting of n locations of interest that we call *targets* and that will be indicated by the set $T = \{t_1, t_2, \ldots, t_n\}$. The targets represent locations that might be under risk of attack and that must be kept under surveillance. Our discrete representation abstracts away from other possible non-target locations and from the topology with which target and non-target locations are connected. Instead, we assume that given two (not necessarily different) targets t_i and t_j there always exists at least one path connecting them, that any path between them can be traveled in both directions, and that the shortest path between them has a temporal traveling cost known in advance and indicated as $d_{ij} \in \mathbb{R}_0^+$ in the following.

The *value* of a target t_i, a quantity proportional to its importance, is denoted as $v_i \in \mathbb{R}^+$ while the target's resilience to attacks is given by the *attack time* $a_i \in \mathbb{R}^+$. The attack time measures the temporal cost that an attacker must spend to successfully compromise the target's security. While it is reasonable to assume that most valuable targets have the highest attack times, we do not make any a priori assumption regarding this aspect.

In this context, a patrolling task is carried out by a single mobile robot/agent traveling from target to target. We assume that the robot has the capability of detecting the presence of

an ongoing attack and undertaking some action to potentially stop it (for example, raising an alarm or alerting a human). Such a capability is localized to the target currently visited by the robot. Thus, if the robot visits target t_i at time τ and an attack on that target has started at a time within the interval $[\max\{0, \tau - a_i\}, \tau)$, then the attack is detected and neutralized. The *status* of a target is *protected* if the patroller is located in it and it becomes *unprotected* when the patroller leaves it.

The threat we assume to face is modeled as coming from an *attacker* agent that, at any time τ, can start an attack to a target t_i. Once the attack is started, the time window $[\tau, \tau + a_i)$ represents an *exposure interval* during which the attacker can be stopped if the patrolling robot visits t_i. As commonly done in security games, we assume an underlying constant-sum interaction between the patroller and the attacker. More precisely, in the case where the patroller stops an attack on target t_i it receives a utility of $U = \sum_{i=1}^{n} v_i$ while the attacker gets 0. On the contrary, if the attack on target t_i is successfully completed the patroller receives a utility of $U - v_i$ while the attacker will conquer the target's value v_i. The patroller's movements in the environment are defined by a Markov chain process where a state represents the currently visited target. The $n \times n$ transition matrix $\mathbf{P}$, where the entry p_{ij} represents the probability of transitioning from target t_i to target t_j, defines the patrolling strategy used to protect the environment. In the following, we shall assume that $\mathbf{P}$ is irreducible and aperiodic [21], [18].

With respect to the classic literature on security games, we propose two relaxations (introduced in [6], here extended with observation errors) that allow us to capture some realistic aspects encountered in real-world applications. The first is about the patroller's movement model. The mainstream approach prescribes that temporal traveling costs of shortest paths (in our setting defined as d_{ij}s) should always correspond to the actual times spent by the patroller for moving between targets. Indeed, in standard strategical settings it can be relatively easy to show how following such rationale weakly dominates the opposite. In our model, we embrace only this basic requirement:

a) we interpret d_{ij} as a lower bound for the time spent traveling between t_i and t_j allowing for occurrences where the patroller takes some extra additional time to transfer.

As we will show, this feature can be useful in scenarios when dealing with a non-fully informed attacker.

The second relaxation we introduce involves the model of the attacker. Typically the attacker is modeled as a rational and fully informed agent that has access to the environment topology, the patrolling strategy being executed by the robot, and its current position. The derived game is hence solved according to a leader-follower paradigm where the attacker substantially best responds to the observed patrolling strategy and the patroller's current position. In our model we instead assume an attacker that is still rational, but that is not fully informed. More specifically, our attacker model is characterized by the following features:

b) the environment topology and, as a consequence, the values of v_i and d_{ij} for all t_i, t_j are not known and not accessible;

c) the patrolling robot cannot be observed while it executes its task in any location of the environment, meaning that the current position is, in general, unknown and no observation-induced belief over the patrolling strategy can be maintained;

d) the attacker[1] is hidden and ready to attack at an unknown target where it can gather local observations under the assumptions described below.

When observing a target during a time where it is *unprotected*, the collected information will not be affected by errors. In other words, we assume a null false-positives rate $\alpha = P(protected \mid unprotected) = 0$. On the contrary, if the attacker is observing a target whose state is *protected*, with probability β it will not detect the presence of the patroller independently of how long the patroller stays on that target an it will be mislead into believing that the target has been *unprotected* for the whole time. In other words, we assume a non-null false-negative rate $\beta = P(unprotected \mid protected) > 0$.

With the model *a)–d)*, we relax some of the assumptions made in security games, namely, that the patrolling setting is fully observable. Instead, the attacker model we consider does not have any prior knowledge on the patrolling setting but only relies on locally limited and noisy observations of a single target. These features capture realistic settings where the planning activities of an attacker take place locally to the target itself and, at the same time, the context in which the patroller is operating (its current position, the set of targets, and the patrolling strategy) are out of reach due to inaccessibility or high intelligence costs. As a concrete example consider a large site protected by an autonomous patrolling unit. Our attacker does not have enough power to monitor the whole site for a long time because it would take too much effort. As a consequence it cannot derive the environment discretization used by the patroller and the surveillance strategy. What it can do, instead, is to loiter at the chosen target area and evaluate its chances on the basis of what it observes there.

IV. Patrolling against a local observer

The scenario described above induces a situation where the patrolling agent travels from target to target by following a transition matrix $\mathbf{P}$. The attacker, hidden at an unknown target that we denote as t_j, observes a sequence of state changes on that target: from *uncovered* to *protected* as soon as the patroller visits t_j and the opposite when it leaves (up to false negatives). Since the success or failure of an attack depends on the patroller's visit within an exposure interval, the attacker is incentivized to log state changes with a timestamp and to extract a time-series defined as subsequent realizations of a random variable R_j modeling

[1]Despite we will generically refer to a single attacker, our model would work also under the interpretation of up to n equally defined and uncoordinated attackers.

the patroller's return time (or inter-arrival time) to target t_j. In the long run, the attacker will take advantage of such knowledge by deriving a belief on $P[R_j > a_j]$, that is the probability that the target will stay uncovered for enough time to complete an attack. In short, we shall call it *attack probability*. Due to assumptions *a)* and *b)*, no inference on the environment topology can be exploited. Specifically, notice that assumption *a)* also applies to self loops, allowing the patroller to leave t_j and then returning to it after an arbitrarily small amount of time. For such reason, the attacker does not have incentives in waiting some extra time after the target has become uncovered, meaning that, if the attack must start, it will initiate as soon as the patroller has left the target (thus justifying the use of R_j).

The problem we face when computing a patrolling strategy for such a local observer is a standard constant-sum setting, namely finding $\mathbf{P}$ such that the maximum attack probability is minimized. Recall that, due to assumption *d)*, the target at which the attacker is hidden is unknown. Because of the difficulty in computing the exact probability, in [6] we provided a first, approximate answer to this question by using the upper bound given by Markov's inequality [17]

$$P[R_j > a_j] \leq \frac{\mathbb{E}[R_j]}{a_j} = U_j$$

and in this work we use this same solution[2].

The objective we seek for the patrolling task is twofold. From one side the patrolling strategy should provide the maximum protection at convergence. That is, it must optimize the attack probability when working under the condition in which the attacker has managed to derive a correct belief over it. At the same time, it is desirable that such condition is hard to meet by the attacker, making the construction of a belief from the observations of R_j as difficult as possible.

One first method that we introduced in [6] is based on the idea of decoupling spatial and temporal decisions when patrolling the environment. This is achieved by an iterative two-step decision process. Let us suppose that the current target occupied by the patroller is t_i. In the first step the next target t_j is selected according to a Markov chain strategy expressed by $\mathbf{P}$. In the second step the patroller draws a value δ_{ij} from the uniform distribution $\mathcal{U}[d_{ij}, d_{ij} + \Delta]$ where $\Delta \in \mathbb{R}_0^+$ is a parameter representing the maximum delay to be applied. Finally, the patroller constraints itself to spend δ_{ij} to reach t_j starting from t_i. In [6] we have shown that the random delay causes the attacker to see a sequence of return times that have the characteristics of white noise. This strategy is summarized in Algorithm 1, where with $\boldsymbol{\pi} = (\pi_1, \pi_2, \ldots, \pi_n)$ we denote the stationary distribution induced by $\mathbf{P}$. Notice that since we assumed that $\mathbf{P}$ is irreducible and aperiodic, $\boldsymbol{\pi}$ is unique and guaranteed to exist.

When adopting a strategy of the form described in Algorithm 1 the upper bound on the attack probability can be written as

[2]A numerical algorithm iteratively converging to the correct attack probability is the subject of ongoing research.

Input: T, distances d_{ij}s, values $v_1, v_2, \ldots, v_n$, attack times $a_1, a_2, \ldots, a_n$, Δ, transition matrix $\mathbf{P}$;
Select the start target $t_i \sim \boldsymbol{\pi}$;
while true do
  Select the next target t_j with probability p_{ij};
  Generate a random time $\delta_{ij} \sim \mathcal{U}[d_{ij}, d_{ij} + \Delta]$;
  Move to t_j spending time δ_{ij};
  $t_i \leftarrow t_j$;

Algorithm 1: Time-delayed patrolling strategy.

$$U_j = \frac{1}{\pi_j a_j} \sum_{i=1}^{n} \pi_i \sum_{h=1}^{n} p_{ij} \frac{2d_{ih} + \Delta}{2}$$

and solving for the optimal stationary distribution $\boldsymbol{\pi}^*$, that is the one that minimizes the maximum upper bound of the attack probability, gives the following (see [6] for details):

$$\pi_i^* = \frac{\mu_i}{\sum_{j=1}^{n} \mu_j} \ \forall t_i \in T, \text{ with } \mu_i = \frac{v_i}{a_i}$$

Given the stationary distribution $\boldsymbol{\pi}^*$, a time-delayed patrolling strategy can be implemented by running the Metropolis-Hastings algorithm to obtain $\mathbf{P}$ (see details in the next section). The rationale behind this strategy is that by tuning the parameter Δ we can lower the autocorrelation in the time series of inter-arrival times observed by the attacker, hence making it harder to forecast.

V. Time-Variant Strategies

An additional way to increase the difficulty of forecasting inter-arrival times is to introduce time-variance in the transition matrix derived with the approach described in the previous section. We introduce time-variant patrolling strategies which are obtained by changing the matrix $\mathbf{P}$ used to determine which vertex to visit next. The key fact to implement this approach is that we can determine an infinite number of transition matrices $\mathbf{P}$ all having the same optimal stationary distribution $\boldsymbol{\pi}^*$. To see this, we just use the Metropolis-Hastings algorithm [23] with a random proposal. More precisely, for a given $\boldsymbol{\pi}^*$ to generate $\mathbf{P}$ we start with a random[3] transition matrix $\mathbf{Q}$ whose entries are all strictly positive. Let $q_{i,j}$ be the (i, j) entry of such matrix, and let

$$\alpha_{i,j} = \min\left\{1, \frac{\pi_j^* q_{j,i}}{\pi_i^*, q_{i,j}}\right\}$$

Then, for $i \neq j$ we set $p_{i,j} = q_{i,j}\alpha_{i,j}$, and p_{ii} is set so that all rows add up to one.

From this premise, we consider four different methods, each defined by the rule used to decide when to drop the current transition matrix $\mathbf{P}^{(k)}$ and switch to a new matrix $\mathbf{P}^{(k+1)}$. Furthermore, we introduce the concept of *duration*

[3]This can be easily created by repeatedly calling a uniform random number generator with strictly positive support to fill up the matrix, and then normalizing each row to add up to one.

of a transition matrix, a quantity that determines the time span during which the same transition matrix is used.

In the **constant-transition** strategy the duration of a transition matrix is defined as a constant number of transitions, say M. That is to say that the patroller keeps an internal counter that is increased every time a new target is visited. When the counter reaches M, a new transition matrix is generated and the counter is reset. Evidently, M is a parameter that needs to be picked upfront. The **random-transition** strategy is a modification of the previous strategy where the duration of each transition matrix is not given by a constant number of transitions, but is rather distributed according to a Poisson distribution with parameter λ. Every time a new transition matrix is generated, its duration is sampled from the Poisson distribution. In other words, each transition matrix is associated with its own randomly defined duration. The **constant-time** strategy relates the duration of a transition matrix to the time spent by the patroller to move from vertex to vertex (differently from the previous two methods that use the number of transitions to determine when to change transition matrix). Specifically, when moving from t_i to t_j the patroller spends time δ_{ij} (recall Algorithm 1). In the constant time approach, a fixed threshold T_{MAX} is defined, and as the patroller moves from vertex to vertex a cumulative variable T is used to add all the various δ_{ij}. When T exceeds T_{MAX}, a new transition matrix is generated and the variable T is reset to 0. In this strategy T_{MAX} is a constant parameter playing a role similar to M in the constant transition strategy. Finally the **random-time** strategy works exactly as the previous one, except that T_{MAX} is not a constant, but is rather sampled from an exponential distribution with parameter ψ[4]. Similarly to the random transition strategy described above, every time a new transition matrix is generated, its duration is determined sampling from the exponential distribution and therefore every transition matrix has a different duration. The time-variant strategies are summarized in Algorithm 2.

Input: T, distances d_{ij}s, values $v_1, v_2, \ldots, v_n$, attack times $a_1, a_2, \ldots, a_n$, Δ, stationary distribution $\boldsymbol{\pi}$;
$k \leftarrow 0$ **while true do**
  Generate $\mathbf{P}^k$ from $\boldsymbol{\pi}$;
  while *the duration of* $\mathbf{P}^k$ *has not expired* **do**
    Set $\mathbf{P} \leftarrow \mathbf{P}^k$ and run Algorithm 1;
  $k \leftarrow k + 1$

Algorithm 2: Time-variant patrolling strategy.

All the above strategies depend on a parameter defining when to change the transition matrix. This choice could be made in different ways. One approach, embraced in the next section, is to perform a preliminary search for a given set of benchmark graphs, and then pick the value of the parameter that maximizes a performance function (see later discussion about possible metrics). This method is off-line, i.e., the parameter is decided upfront and remains constant throughout the run. In the conclusions we will discuss another possible on-line approach where the decision on when to change matrix is rather taken on the fly.

[4]For the exponential distribution we use the parametrization where ψ is the expectation.

VI. Evaluation

We generated 50 environments of different sizes considering complete graphs with 10, 20, 30, 40, and 50 targets, and, for each size, we generated 10 random instances. The graphs vertices were created by randomly placing points on a 100×100 plane and then setting each edge's temporal cost $d_{i,j}$ to the Euclidean distance between t_i and t_j. The value of each target v_i was drawn from $1 + \mathcal{U}[0, 10]$ while the attack time was drawn from $\mathcal{U}[D, 3D]$ where D is the average of the temporal costs $d_{i,j}$.

While in [6] we considered a single strategy for the attacker, in this paper we test our algorithm against three different types of observing attackers. In each case the attacker computes a prediction of the next inter-arrival time at the target as soon as a transition from *protected* to *unprotected* is observed. If the predicted value is greater than the target's attack time, the attack is attempted. Starting from its local observations, each type of attacker uses a different method to predict the time when the attacker will visit the vertex again.

The first type of attacker assumes that inter-arrival times at target t_i follow an exponential distribution with parameter λ_i and uses a maximum likelihood approach to estimate such parameter. The attacker then derives a prediction for the next inter-arrival time by taking $1/\lambda_i$, that is the expectation of the hypothesized exponential distribution.

The second type of attacker uses a nearest neighbor (NN) approach to forecast a time series [9]. The attacker observing target t_i acquires a sequence of observations $O = (R_i^0, R_i^1, R_i^2, \ldots)$ for the inter-arrival times. It then considers all the sub-sequences of length m, where m can be thought as the attacker's finite memory, and computes the distance between each of these sequences and the last m inter-arrival times that have been observed. The closest sub-sequence (different from the last one) is selected and the inter-arrival time observed immediately after such sub-sequence is taken as the prediction. In all our experiments, $m = 10$.

The last type of attacker exploits a deep neural network (DNN) consisting of a fully connected regression network with 200 hidden units using a long short-memory layer [13]. Owing to the necessity of having enough training data, this method uses the first 500 observations to train the network and then makes predictions for the times of the remaining visits. The parameter 500 was obtained after manual tuning, and never represents more than 90% of the data available for training and testing. It shall be noted that, owing to the large number of parameters to tune, this method requires significantly more data than the other two and from a practical standpoint one could argue that when observations have a cost, an attacker would be unlikely to use it. Nonetheless, due to the vast popularity of these methods, it is interesting to use this approach to assess intrinsic strengths or weaknesses of our patrolling strategy.

We define an evaluation metric denoted as *protection ratio*. This quantity is obtained empirically, by simulating a deployment of the patrolling strategy we want to evaluate. First we simulate the strategy for a total of K transitions. (In our experiments, K was set to 10000 for the first two methods and to 100000 for the DNN, due to the necessity of generating sufficient training data). Afterwards, for each target t_i we consider K_i as the set of transitions after which the status of t_i changed from *protected* to *unprotected*. For each $k \in K_i$ we run the three attacker types feeding them with the whole history of inter-arrival times generated at t_i by all the transitions preceding k. We then record each attacker's decision and, if that decision was to start an attack, we label the attack as successful/unsuccessful by looking at how the patrolling mission would unfold ahead of k. Call $N_A(i)$ and $N_A^u(i)$ the number of attack attempts and the number of unsuccessful attacks generated by such procedure on target t_i, respectively. The protection ratio for target t_i is then defined as $v_i N_A^u(i)/N^A(i)$. Ideally, this metric should be high, i.e., for high-value targets we prefer patrolling strategies causing the attacker to perform many unsuccessful attacks. In the experiments we present, the choice for the parameters defining the four time-variant strategies (shown at the top of the figures) were informed by a preliminary comparative evaluation of different choices. In particular, we run the algorithms on a set of benchmark problems and picked the values giving the best performance averaged across all instances. Also, the plots we show (obtained in instances with 30 targets) are a representative selection of the results obtained for the other cases that, for reasons of space and clarity, we do not show.

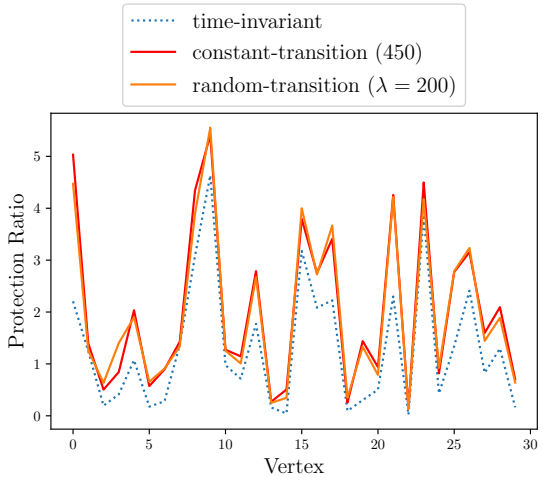

Fig. 1. Protection ratio for each vertex with attacker using a NN strategy.

In Figures 1 and 2 we start by contrasting the time-variant strategies with the time-invariant method we proposed in [6]. In particular, we plot the protection ratio for one of the environments with 30 vertices. In this case the attacker used the NN approach. As it can be seen, the temporal variant strategies outperform the time-invariant strategy at the majority of targets ensuring a more extensive protection.

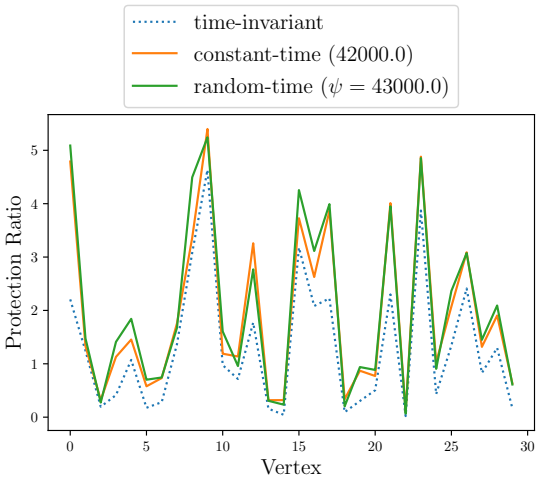

Fig. 2. Protection ratio for each vertex with attacker using a NN strategy.

Vertices where the gaps are marginal are those typically characterized by small attack times. For such vertices, the patroller can do little to prevent an attack since they represent locations intrinsically difficult to protect.

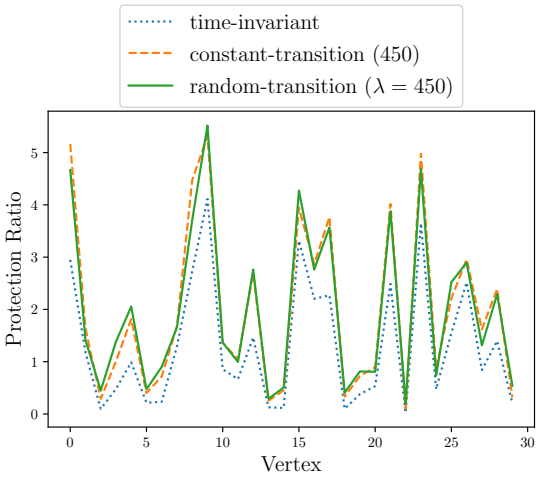

Fig. 3. Protection ratio for each vertex with attacker using a maximum likelihood strategy.

Next, in Figures 3 and 4 we display analog results, by considering the case where the attacker uses the maximum likelihood method. It can be observed that the performance is rather similar, with time-variance outperforming the non-variant strategy. A possible explanation behind such trends is that time-variance makes these patrolling strategies *harder to learn* and hence predicting inter-arrival times is likely to be subject to errors. Slightly more complex methods such as NN do not tend to outperform simpler approaches like the prediction based on maximum likelihood (similar results we obtained with number of targets between 10 and 50).

This intuition about the hardness of forecasting these strategies is confirmed by the results obtained when the attacker uses a DNN. This is shown in Figures 5 and 6.

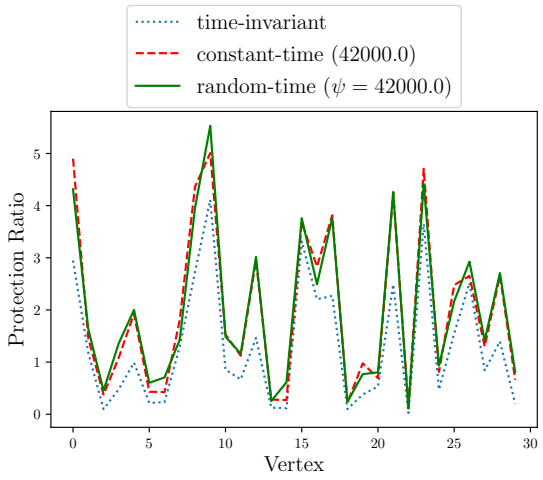

Fig. 4. Protection ratio for each vertex with attacker using a maximum likelihood strategy.

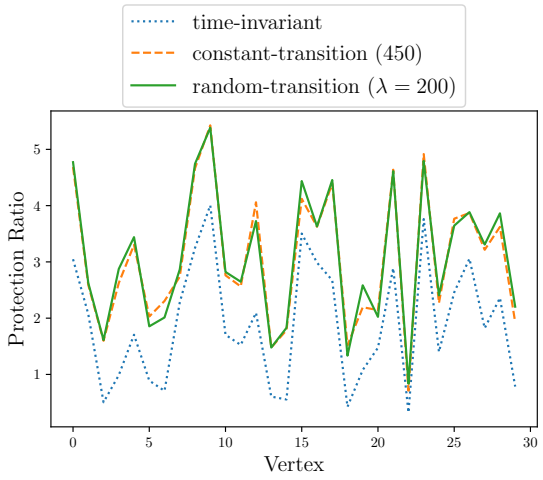

Fig. 5. Protection ratio for each vertex with attacker using a DNN.

First, observe that the absolute values for the protection ratio are essentially comparable to those obtained with the other types of attacker. Therefore even if the attacker uses a more sophisticated technique to forecast the next visit to the vertex, the performance does not improve. We explain this result with the intrinsic hardness of predicting the time series generated by our patrolling strategies. This is particularly relevant because the tests with the DNN were on purpose skewed in favor of the attacker: the network was trained with significantly more data than the samples made available to the attackers based on the NN or maximum likelihood. The only outlier case appears to be the protection ratio for vertex 23 in Figure 6. Due to the *black box* nature of the forecast approach implemented by the DNN, it is difficult to explain why this happens. However, this single outlier does not hinder the overall conclusions we derived, although further investigations will be done in the future. Figure 7 further corroborates our assessment that data intensive methods like DNN do not significantly outperform simple strategies. The

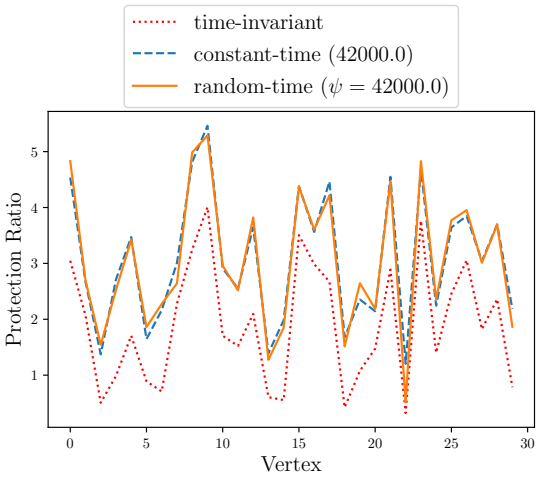

Fig. 6. Protection ratio for each vertex with attacker using a DNN.

figure shows the protection ratio for a testcase graph when the attacker uses the three different strategies we considered. While the maximum likelihood and the NN offer comparable performance, the DNN almost uniformly performs worse that the other two strategies in terms of ability to correctly predict the next arrival time, while at the same time requiring significantly more data to train.

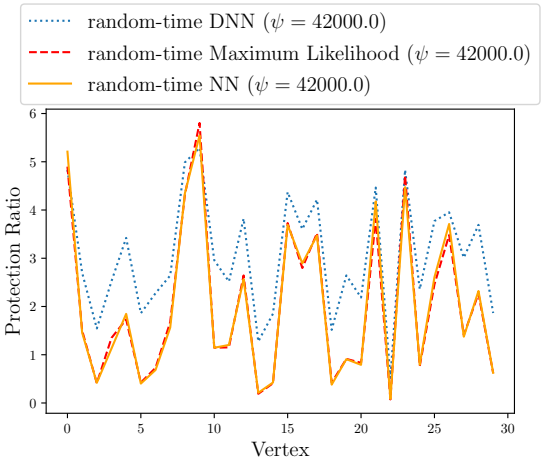

Fig. 7. Comparison between the protection ratio achieved on a testcase graph against three different attack strategies.

In the last experiment we present, we assess the performance of our strategies with respect to the false negatives rate β. We analyze how the average protection ratio varies with respect to what we denote as *observation accuracy*, defined as $1 - \beta$. The lower this accuracy, the more frequently the attacker will believe that the target has been unattended even if the patroller has actually been there to check it.

In Figures 8 and 9, we report this analysis for the random-time strategy and for the time-invariant one against the maximum likelihood attacker. The intuitive trend that would be expected from an attacker affected by increasing levels of observation noise is that of an increase in the protection ratio.

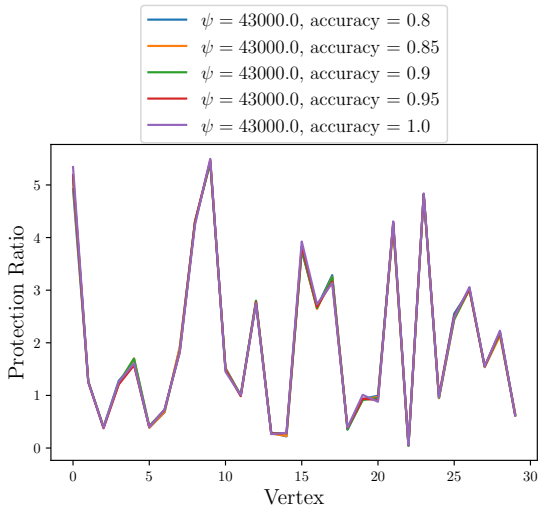

Fig. 8. Protection ratio for each vertex with the random-time strategy and the maximum likelihood attacker for different observation accuracy.

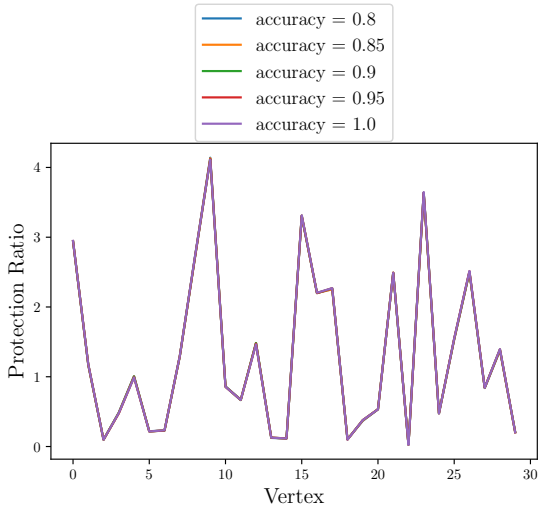

Fig. 9. Protection ratio for each vertex with the time-invariant strategy and the maximum likelihood attacker with different observation accuracy.

That is, attackers with higher false negatives rates should be, on average, performing worse against our patroller. However, the figures reported below draw a less intuitive and more insightful picture. We notice that for not remarkably low, and hence reasonable, values of accuracy (≥ 0.8) no significant difference can be observed in the average protection ratio for both strategies.

VII. Conclusions and Future Work

We studied an adversarial patrolling setting where the attacker is characterized by an observation model allowing it to gather information only locally to a target. We proposed heuristic methods to generate patrolling strategies that, using delays and time-variance, provide protection and are difficult to forecast. A compelling future direction for the present work is the study of on-line techniques to adapt time-variance. One possible approach is to have the patroller running an online model of a baseline attacker (e.g., one that runs a maximum likelihood estimator) and use it in the decision of when to switch from a strategy to another. This extension is left for future work.

References

[1] P. Agharkar, R. Patel, and F. Bullo. Robotic surveillance and Markov chains with minimal first passage time. In *Proc. CDC*, pages 6603–6608, 2014.

[2] N. Agmon, V. Sadov, G. A Kaminka, and S. Kraus. The impact of adversarial knowledge on adversarial planning in perimeter patrol. In *Proc. AAMAS*, pages 55–62, 2008.

[3] B. An, M. Brown, Y. Vorobeychik, and M. Tambe. Security games with surveillance cost and optimal timing of attack execution. In *Proc. AAMAS*, pages 223–230, 2013.

[4] B. An, D. Kempe, C. Kiekintveld, E. Shieh, S. Singh, M. Tambe, and Y. Vorobeychik. Security games with limited surveillance. In *Proc. AAAI*, pages 1241–1248, 2012.

[5] A. B. Asghar and S. L Smith. Stochastic patrolling in adversarial settings. In *Proc. ACC*, pages 6435–6440. IEEE, 2016.

[6] N. Basilico and S. Carpin. Balancing unpredictability and coverage in adversarial patrolling settings. In *WAFR*, 2018.

[7] N. Basilico, N. Gatti, and F. Amigoni. Patrolling security games: Definition and algorithms for solving large instances with single patroller and single intruder. *ARTIF INTELL*, 184:78–123, 2012.

[8] A. Blum, N. Haghtalab, and A.D. Procaccia. Lazy defenders are almost optimal against diligent attackers. In *Proc. AAAI*, pages 573–579, 2014.

[9] G. Bontempi, S. Ben Taieb, and Y. Le Borgne. Machine learning strategies for time series forecasting. In *Business Intelligence*, pages 62–77, 2013.

[10] Y. Chevaleyre. Theoretical analysis of the multi-agent patrolling problem. In *Proc. IAT*, pages 302–308, 2004.

[11] V. Conitzer and T. Sandholm. Computing the optimal strategy to commit to. In *Proc. EC*, pages 82–90, 2006.

[12] J. Grace and J. Baillieul. Stochastic strategies for autonomous robotic surveillance. In *Proc. CDC*, pages 2200–2205, 2005.

[13] S. Hochreiter and J. Schmidhuber. Long short-term memory. *NEURAL COMPUT*, 9:1735–1780, 1997.

[14] David Klaška, Antonín Kučera, Tomáš Lamser, and Vojtěch Řehák. Automatic synthesis of efficient regular strategies in adversarial patrolling games. In *Proc. AAMAS*, pages 659–666, 2018.

[15] Efrat Sless Lin, Noa Agmon, and Sarit Kraus. Multi-robot adversarial patrolling: Handling sequential attacks. *ARTIF INTELL*, 274:1–25, 2019.

[16] S Meghana, Teja V Nikhil, Raghuveer Murali, S Sanjana, R Vidhya, and Khurram J Mohammed. Design and implementation of surveillance robot for outdoor security. In *Proc. RTEICT*, pages 1679–1682, 2017.

[17] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

[18] A. Papoulis and S.U. Pillai. *Probability, Random Variables, and Stochastic Processes*. McGraw-Hill, 4th edition, 2002.

[19] J. Pita, M. Jain, M. Tambe, F. Ordóñez, and S. Kraus. Robust solutions to stackelberg games: Addressing bounded rationality and limited observations in human cognition. *ARTIF INTELL*, 174(15):1142–1171, 2010.

[20] D. Portugal and R. Rocha. A survey on multi-robot patrolling algorithms. In *Proc. DoCEIS*, pages 139–146, 2011.

[21] N. Privault. *Understanding Markov Chains*. Springer, 2013.

[22] C. Robin and S. Lacroix. Multi-robot target detection and tracking: taxonomy and survey. *AUTON ROBOT*, 40(4):729–760, 2016.

[23] S.M. Ross. *Introduction to Probability Models*. Elsevier, 2014.

[24] M. Tambe. *Security and game theory: algorithms, deployed systems, lessons learned*. Cambridge University Press, 2011.

[25] H Xu, AX Jiang, A Sinha, Z Rabinovich, S Dughmi, and M Tambe. Security games with information leakage: modeling and computation (2015). arxiv preprint. *Proc. IJCAI*, pages 674–680, 2015.

[26] Z.u Yin, M. Jain, M. Tambe, and F. Ordonez. Risk-averse strategies for security games with execution and observational uncertainty. In *Proc. AAAI*, pages 758–763, 2011.